IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Bryan M. Cantrill

Application No.: 10/699,062	Confirmation No.: 2597
Filed: October 31, 2003	Art Unit: 2162
For: MECHANISM FOR DATA AGGREGATION IN A TRACING FRAMEWORK	Examiner: D. Y. Myint

DECLARATION PURSUANT TO 37 C.F.R. § 1.131

In connection with Applicant's Response to the Office Action issued on May 4, 2006, this declaration sets forth the pertinent facts proving conception and actual reduction to practice of the claimed invention prior to **May 16, 2002**.

1. I am the sole inventor listed on U.S. Patent Application Serial No. 10/699,062 entitled "MECHANISM FOR DATA AGGREGATION IN A TRACING FRAMEWORK" filed on October 31, 2003.

2. I conceived and completed the actual reduction to practice of the claimed invention at least prior to March 12, 2002, when I gave an internal company speech directed, in part, to the claimed invention. The speech, which was conducted on March 12, 2002, included a slide presentation and a live demonstration of the claimed invention. A copy of relevant portions of the slide presentation entitled "DTrace: Dynamic Tracing For Solaris" dated March 11, 2002 (*see* Presentation, slide 1, slides 66-73) is included under tab 1. Further, a DVD video of the speech showing the live

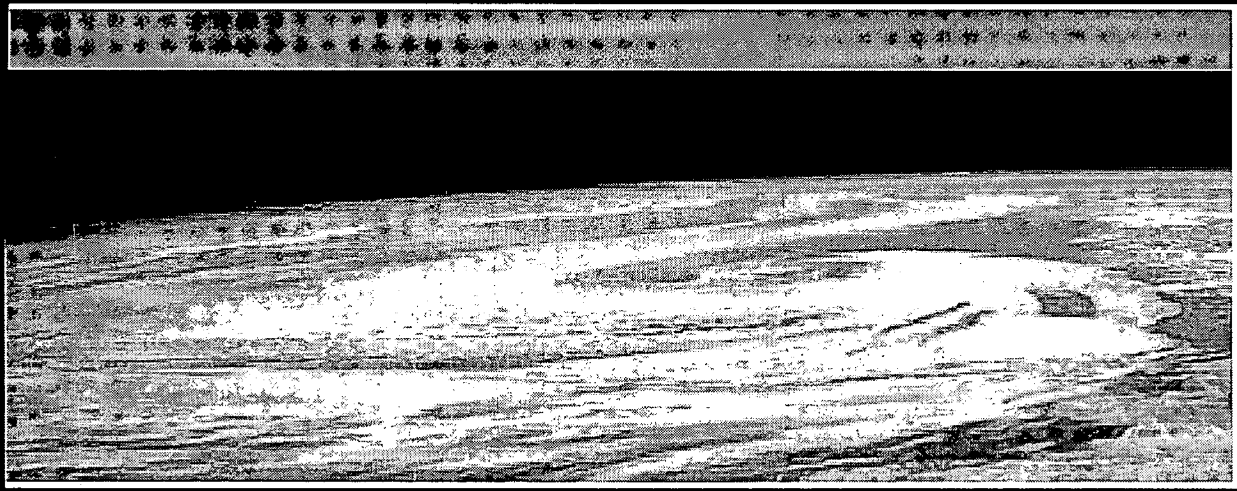
demonstration is attached under tab 2. The relevant portions of the DVD include, at least, chapters 27-30 of Part 1.

I, Bryan M. Cantrill, hereby declare that all statements made herein of my own knowledge are true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signed this day 5, of September.

[illegible] for Bryan M. Cantrill w̄ permission

Bryan M. Cantrill



DTrace: Dynamic Tracing For Solaris

Bryan Cantrill (bmc@eng)   Mike Shapiro (mws@eng)

Solaris Kernel Technologies

Sun microsystems

Aggregations

- An *aggregating function* is a function $f(x)$, where x is a sequence of arbitrary length, for which there exists an aggregating function $f'(x)$ such that:

$$f'(f(x_0), f(x_1), \dots f(x_n)) = f(x_0, x_1, \dots x_n)$$

- E.g., COUNT, MEAN, MAXIMUM, and MINIMUM are aggregating functions; MEDIAN, and MODE are not

Sun microsystems

Aggregations, cont.

- When data is to be processed using an aggregating function, the implementation can be made very efficient:
  - Trace records need not be generated; only the intermediate results from the aggregating function need to be stored
  - Intermediate results from aggregating functions can be stored *per CPU*, thereby eliminating data sharing
  - Aggregating function can be periodically performed on all per CPU intermediate results to derive system-wide result

Aggregations, cont.

- An *aggregation* is an associative table keyed by an n-tuple where each value is the result of an aggregating function
- n-tuple consists of a list of D expressions
- Aggregating functions are provided by the framework
- Framework provides a single aggregation per consumer

Sun microsystems

Aggregations, cont.

- Current aggregating functions:
  - MAX(*expr*): the intermediate result is set to the greater of the intermediate result and *expr*
  - COUNT: increments the intermediate result
  - QUANTIZE(*expr*): the intermediate result consists of 64 power-of-two buckets; the bucket corresponding to *expr* is incremented
  - AVG(*expr*): the intermediate result consists of a count and a total; the count is incremented and the total is increased by *expr*

Sun microsystems

Aggregation Example

- For example, maximum kernel **`bcopy`**() size by command name:
  - Enable probe with function "**`bcopy`**", name "**`entry`**"
  - Aggregate on:

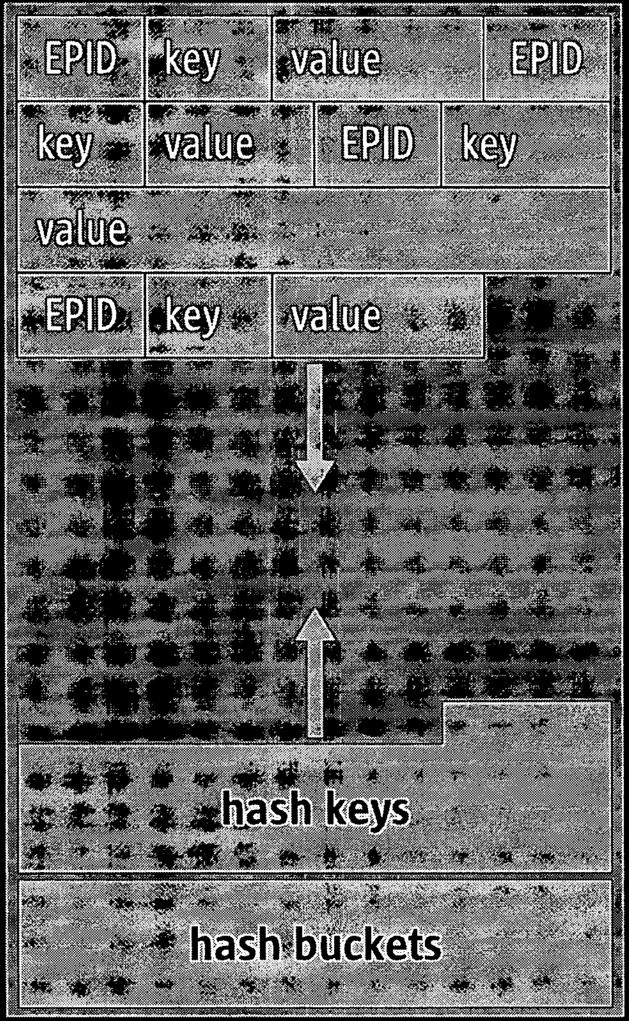
    ```
    curthread->t_procp->p_user.u_comm
    ```
  - Set aggregating function to "**`max(arg2)`**"

Sun microsystems

Aggregation Implementation

- Aggregations are implemented using the same buffer infrastructure as trace buffers
- Buffer switching and copying thus fall out
- Aggregations are an associative table; buffering is complicated by the presence of hash table metadata

Aggregation Implementation



- *Data* grows from start of buffer
- *Metadata* grows from end of buffer
- *Only* data is copied out
- EPID is in data record, but is *not* considered to be part of the key

Sun microsystems

Aggregation Implementation

- Library applies aggregating function to consumed data, using formerly consumed data as intermediate result
- Allows the kernel to discard the metadata contents of consumed aggregation buffers
- Allows drops to be easily eliminated in long-running aggregations